Probabilistic Combination of Classifier and Cluster Ensembles for Non-transductive Learning

Ayan Acharya* Eduardo R. Hruschka† Joydeep Ghosh* Badrul Sarwar‡
Jean-David Ruvini‡

Abstract

Unsupervised models can provide supplementary soft constraints to help classify new target data under the assumption that similar objects in the target set are more likely to share the same class label. Such models can also help detect possible differences between training and target distributions, which is useful in applications where concept drift may take place. This paper describes a Bayesian framework that takes as input class labels from existing classifiers (designed based on labeled data from the source domain), as well as cluster labels from a cluster ensemble operating solely on the target data to be classified, and yields a consensus labeling of the target data. This framework is particularly useful when the statistics of the target data drift or change from those of the training data. We also show that the proposed framework is privacy-aware and allows performing distributed learning when data/models have sharing restrictions. Experiments show that our framework can yield superior results to those provided by applying classifier ensembles only.

1 Introduction

In several data mining applications, one builds an initial classification model that needs to be applied to unlabeled data acquired subsequently. Since the statistics of the underlying phenomena being modeled changes with time, these classifiers may also need to be occasionally rebuilt if performance degrades beyond an acceptable level. In such situations, it is desirable that the classifier functions well with as little labeling of new data as possible, since labeling can be expensive in terms of time and money, and a potentially error-prone process. Moreover, the classifier should be able to adapt to changing statistics to some extent, given the aforementioned constraints.

This paper addresses the problem of combining multiple classifiers and clusterers in a fairly general setting, that includes the scenario sketched above. An ensemble of classifiers is first learnt on an initial labeled training dataset after which the training data can be discarded. Subsequently, when new unlabeled target data is encountered, a cluster ensemble is applied to it, thereby generating cluster labels for the target data. The heart of our approach is a Bayesian framework that combines both sources of information (class/cluster labels) to yield a consensus labeling of the target data.

The setting described above is, in principle, different from transductive learning setups where both labeled and unlabeled data are available at the same time for model building [19], as well as online methods [6]. Additional differences from existing approaches are described in the section on related works. For the moment we note that the underlying assumption is that similar new objects in the target set are more likely to share the same class label. Thus, the supplementary constraints provided by the cluster ensemble can be useful for improving the generalization capability of the resulting classifier system. Also, these supplementary constraints can be useful for designing learning methods that help determining differences between training and target distributions, making the overall system more robust against concept drift.

We also show that our approach can combine cluster and classifier ensembles in a privacy-preserving setting. This approach can be useful in a variety of applications. For example, the data sites can represent parties that are a group of banks, with their own sets of customers, who would like to have a better insight into the behavior of the entire customer population without compromising the privacy of their individual customers.

The remainder of the paper is organized as follows. The next section addresses related work. The proposed Bayesian framework — named **BC^{3}E**, from **B**ayesian **C**ombination of **C**lassifiers and **C**lusterer **E**nsembles — is described in Section 3. Issues with privacy preservation are discussed in Section 4 and the experimental results are reported in Section 5. Finally, Section 6 concludes the paper.

2 Related Work

The combination of multiple classifiers to generate an ensemble has been proven to be more useful compared

*University of Texas at Austin, Austin, TX, USA. Email: {aacharya@, ghosh@ece}.utexas.edu

†University of Sao Paulo at Sao Carlos, Brazil. Email: erh@icmc.usp.br

‡eBay Research Lab, San Jose, CA, USA. Email: {bsarwar, jruvini}@ebay.com

to the use of individual classifiers [17]. Analogously, several research efforts have shown that cluster ensembles can improve the quality of results as compared to a single clusterer — *e.g.*, see [21] and references therein. Most of the motivations for combining ensembles of classifiers and clusterers are similar to those that hold for the standalone use of either classifier or cluster ensembles. Additionally, unsupervised models can provide supplementary constraints for classifying new data and thereby improve the generalization capability of the resulting classifier. These successes provide the motivation for designing effective ways of leveraging both classifier and cluster ensembles to solve challenging prediction problems.

Specific mechanisms for combining classification and clustering models however have been introduced only recently in the Bipartite Graph-based Consensus Maximization (**BGCM**) algorithm [13], the Locally Weighted Ensemble (**LWE**) algorithm [12] and, in the **C^{3}E** algorithm [3]. Both **BGCM** and **C^{3}E** have parameters that control the relative importance of classifiers and clusterers. In traditional semi-supervised settings, such parameters can be optimized via cross-validation. However, if the training and the target distributions are different, cross-validation is not possible. From this viewpoint, our approach (**BC^{3}E**) can be seen as an extension of **C^{3}E** [3] that is capable of dealing with this issue in a more principled way. In addition, the algorithms in [13, 12, 3] do not deal with privacy issues, whereas our probabilistic framework can combine class labels with cluster labels under conditions where sharing of individual records across data sites is not permitted. It uses a soft probabilistic notion of privacy, based on a quantifiable information-theoretic formulation [16]. Note that existing works on Bayesian classifier ensembles — *e.g.*, [10, 8, 14] — do not deal with privacy issues.

From the clustering side, the proposed model borrows ideas from the Bayesian Cluster Ensemble [21]. In [1], we introduced some preliminary ideas that are further developed in our current paper. In particular, the algorithm in [1] is not capable of automatically estimating the importance that classifiers and clusterers should have. This property is fundamental for applications where training and target distributions are different. In addition, the Bayesian model presented here is considerably different and requires more sophisticated inference and estimation procedures.

3 Probabilistic Model

We assume that a classifier ensemble has been (previously) induced from a training set. At this point and assuming a non-transductive setting, the training data can be discarded if so desired. Such a classifier ensemble is employed to generate a number of class labels (one from each classifier) for every object in the target set. **BC^{3}E** refines such classifier prediction with the help of a cluster ensemble. Each base clustering algorithm that is part of the ensemble partitions the target set, providing cluster labels for each of its objects. From this point of view, the cluster ensemble provides supplementary constraints for classifying those objects, with the rationale that similar objects — those that are likely to be clustered together across (most of) the partitions that form the cluster ensemble — are more likely to share the same class label.

Consider a target set $\mathcal{X} = \{\mathbf{x}_n\}_{n=1}^{N}$ formed by N unlabeled objects. A classifier ensemble composed of r_1 models has produced r_1 class labels for every object $\mathbf{x}_n \in \mathcal{X}$. It is assumed that the target objects belong to k classes denoted by $C = \{C_i\}_{i=1}^{k}$ and at least one object from each of these classes was observed in the training phase (*i.e.* we do not consider "novel" classes in the target set). Similarly, consider that a cluster ensemble comprised of r_2 clustering algorithms has generated cluster labels for every object in the target set. The number of clusters need not be the same across different clustering algorithms. Also, it should be noted that the cluster labeled as *1* in a given data partition may not align with the cluster numbered *1* in another partition, and none of these clusters may correspond to class *1*. Given the class and cluster labels, the objective is to come up with refined class probability distributions $\{(\hat{P}(C_i|\mathbf{x}_n))_{i=1}^{k} = \boldsymbol{y}_n\}_{n=1}^{N}$ of the target set objects. This framework is illustrated in Fig. 1.

The observed class and cluster labels are represented as $\boldsymbol{W} = \{\{\boldsymbol{w}_{1nl}\}, \{\boldsymbol{w}_{2nm}\}\}$ where $\boldsymbol{w}_{1nl}$ is the 1-of-k representation of class label of the n^{th} object given by the l^{th} classifier, and $\boldsymbol{w}_{2nm}$ is the 1-of-$k^{(m)}$ representation of cluster label assigned to the n^{th} object by the m^{th} clusterer. A generative model is proposed to explain the observations $\boldsymbol{W}$, where each object $\mathbf{x}_n$ has an underlying mixed-membership to the k different classes. Let $f(\boldsymbol{y}_n)$ denote the latent mixed-membership vector for $\mathbf{x}_n$, where $f(\mathbf{x}) = \frac{\exp(x_i)}{\sum_{i=1}\exp(x_i)}$ is the softmax function. $\boldsymbol{y}_n$ is sampled from a normal distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Also, corresponding to the i^{th} class and m^{th} base clustering, we assume a multinomial distribution $\boldsymbol{\beta}_{mi}$ over the cluster labels of the m^{th} base clustering. Therefore, $\boldsymbol{\beta}_{mi}$ is of dimension $k^{(m)}$ and $\sum_{j=1}^{k^{(m)}} \boldsymbol{\beta}_{mij} = 1$ if the m^{th} base clustering has $k^{(m)}$ clusters. The data generative process, whose corresponding graphical model is shown in 2, can be summarized as follows.

For each $\mathbf{x}_n \in \mathcal{X}$:

1. Choose $\boldsymbol{y}_n \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} \in \mathbb{R}^k$ is the mean

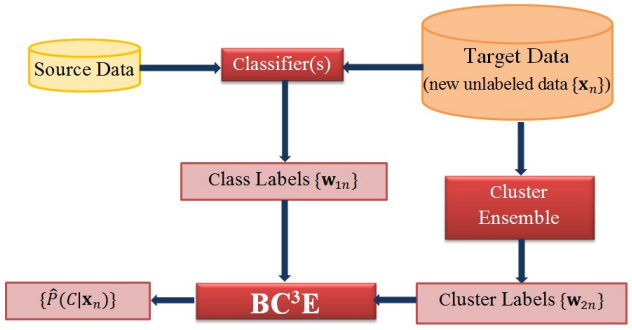

Figure 1: Combining Classifiers and Clusterers.

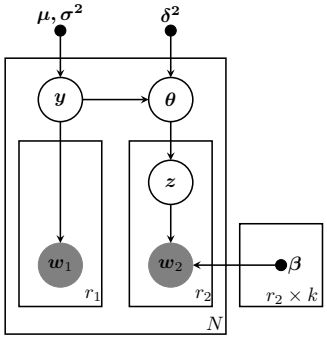

Figure 2: Graphical Model for **BC³E**

and $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ is the covariance.

2. Choose $\boldsymbol{\theta}_n \sim \mathcal{N}(\boldsymbol{y}_n, \delta^2 I_k)$, where $\delta^2 \geq 0$ is the scaling factor of the covariance of the normal distribution centered at $\boldsymbol{y}_n$, and I_k is the identity $k \times k$ matrix.
3. $\forall l \in \{1, 2, \cdots, r_1\}$, choose $\boldsymbol{w}_{1nl} \sim f(\boldsymbol{y}_n)$.
4. $\forall m \in \{1, 2, \cdots, r_2\}$:
   (a) Choose $\boldsymbol{z}_{nm} \sim f(\boldsymbol{\theta}_n)$, where $\boldsymbol{z}_{nm}$ is a k-dimensional vector with 1-of-k representation.
   (b) Choose $\boldsymbol{w}_{2nm} \sim$ multinomial$(\boldsymbol{\beta}_{r\boldsymbol{z}_{nm}})$.

The observed class labels $\{\boldsymbol{w}_{1nl}\}$ are assumed to be sampled from the latent mixed-membership vector $f(\boldsymbol{y}_n)$. If the n^{th} object is sampled from the i^{th} class in the m^{th} base clustering (implying $z_{nmi} = 1$), then its cluster label will be sampled from the multinomial distribution $\boldsymbol{\beta}_{mi}$. This particular generative process is analogous to the one used by the Bayesian Cluster Ensemble in [21]. The fact that $\boldsymbol{\theta}_n$ is sampled from $\mathcal{N}(\boldsymbol{y}_n, \delta^2 I_k)$ needs further clarification. In practice, the observed class labels and cluster labels carry different intrinsic weights. If the observations from the classifiers are assigned too much weight compared to those from clustering, there is little hope for the clustering to enhance classification. Similarly, if the observations from the clustering are given too much of importance, the classification performance might deteriorate. Ideally, the unsupervised information is only expected to enhance the classification accuracy.

Aimed at building a "safe" model that can intelligently utilize or reject the unsupervised information, $\boldsymbol{\theta}_n$ is sampled from $\mathcal{N}(\boldsymbol{y}_n, \delta^2 I_k)$ where the parameter δ decides how much the observations from the clusterings can be trusted. If δ^2 is a large positive number, $\boldsymbol{y}_n$ does not have to explain the posterior of $\boldsymbol{\theta}_n$. From the generative model perspective, this means that the sampled value of $\boldsymbol{\theta}_n$ is not governed by $\boldsymbol{y}_n$ anymore as the distribution has very large variance. On the other hand, if δ^2 is a small positive number, $\boldsymbol{y}_n$ has to explain the posterior of $\boldsymbol{\theta}_n$ and hence the observations from the clustering. Therefore, the posteriors of $\{\boldsymbol{y}_n\}$ are expected to get more accurate compared to the case if they only had to explain the classification results. A concrete quantitative argument for this intuitive statement will be presented later.

To address the log-likelihood function of **BC³E**, let us denote the set of hidden variables by $\boldsymbol{Z} = \{\{\boldsymbol{y}_n, \{\boldsymbol{\theta}_n\}, \{\boldsymbol{z}_{nm}\}\}$. The model parameters can conveniently be represented by $\boldsymbol{\zeta}_0 = \{\boldsymbol{\mu}, \mathbf{\Sigma}, \delta^2, \{\boldsymbol{\beta}_{mi}\}\}$. The joint distribution of the hidden and observed variables can be written as:

$$p(\mathbf{X}, \boldsymbol{Z}|\boldsymbol{\zeta}_0) = \prod_{n=1}^{N} p(\boldsymbol{y}_n|\boldsymbol{\mu}, \mathbf{\Sigma}) p(\boldsymbol{\theta}_n|\boldsymbol{y}_n, \delta^2 I_k). \tag{3.1}$$

$$\prod_{l=1}^{r_1} p(w_{1nl}|f(\boldsymbol{y}_n)) \prod_{m=1}^{r_2} p(\boldsymbol{z}_{nm}|f(\boldsymbol{\theta}_n)) p(w_{2nm}|\boldsymbol{\beta}, \boldsymbol{z}_{nm})$$

The inference and estimation is performed using Variational Expectation-Maximization (**VEM**) to avoid computational intractability due to the coupling between $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$.

3.1 Approximate Inference and Estimation:

3.1.1 Inference: To obtain a tractable lower bound on the observed log-likelihood, we specify a fully factorized distribution to approximate the true posterior of the hidden variables:

$$q(\boldsymbol{Z}|\{\boldsymbol{\zeta}_n\}_{n=1}^{N}) = \prod_{n=1}^{N} q(\boldsymbol{y}_n|\boldsymbol{\mu}_n, \mathbf{\Sigma}_n) q(\boldsymbol{\theta}_n|\boldsymbol{\epsilon}_n, \mathbf{\Delta}_n)$$

$$\prod_{m=1}^{r_2} q(\boldsymbol{z}_{nm}|\boldsymbol{\phi}_{nm}) \tag{3.2}$$

where $\boldsymbol{y}_n \sim \mathcal{N}(\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, $\boldsymbol{\theta}_n \sim \mathcal{N}(\boldsymbol{\epsilon}_n, \boldsymbol{\Delta}_n)\ \forall n \in \{1, 2, \cdots, N\}$, $\boldsymbol{z}_{nm} \sim$ multinomial$(\boldsymbol{\phi}_{nm})\ \forall n \in \{1, 2, \cdots, N\}$ and $\forall m \in \{1, 2, \cdots, r_2\}$, and $\boldsymbol{\zeta}_n = \{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n, \boldsymbol{\epsilon}_n, \boldsymbol{\Delta}_n), \{\boldsymbol{\phi}_{nm}\}\}$ – the set of variational parameters corresponding to the n^{th} object. Further, $\boldsymbol{\mu}_n, \boldsymbol{\epsilon}_n \in \mathbb{R}^k$, $\boldsymbol{\Sigma}_n, \boldsymbol{\Delta}_n \in \mathbb{R}^{k \times k}\ \forall n$ and $\boldsymbol{\phi}_{nm} = (\phi_{nmi})_{i=1}^k\ \forall n, m$; where the components of the corresponding vectors are made explicit. To work with less parameters, all the covariance matrices are assumed to be diagonal. Therefore, $\boldsymbol{\Sigma} = \text{diag}\left((\sigma_i)_{i=1}^k\right)$, $\boldsymbol{\Sigma}_n = \text{diag}\left((\sigma_{ni})_{i=1}^k\right)$, and $\boldsymbol{\Delta}_n = \text{diag}\left((\delta_{ni})_{i=1}^k\right)$. Using Jensen's inequality, a lower bound on the observed log-likelihood can be derived as:

$$\begin{aligned} \log[p(\boldsymbol{X}|\boldsymbol{\zeta}_0)] &\geq \mathbf{E}_{q(\boldsymbol{Z})}\left[\log[p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\zeta}_0)]\right] + H(q(\boldsymbol{Z})) \\ &= \mathcal{L}(q(\boldsymbol{Z})) \end{aligned} \tag{3.3}$$

where $H(q(\boldsymbol{Z})) = -\mathbf{E}_{q(\boldsymbol{Z})}[\log[q(\boldsymbol{Z})]]$ is the entropy of the variational distribution $q(\boldsymbol{Z})$, and $\mathbf{E}_{q(\boldsymbol{Z})}[.]$ is the expectation w.r.t $q(\boldsymbol{Z})$.

Let $\mathcal{Q}$ be the set of all distributions having a fully factorized form as given in (3.2). The optimal distribution that produces the tightest possible lower bound $\mathcal{L}$ is given by:

$$q^* = \arg\min_{q \in \mathcal{Q}} \text{KL}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\zeta}_0)||q(\boldsymbol{Z})). \tag{3.4}$$

In equations (4), (6), (8), (10), (12), (13) and (14) in Table 1, the optimal values of the variational parameters that satisfy (3.4) are presented. Since the logistic normal distribution is not conjugate to multinomial, the update equations of all the parameters cannot be obtained in closed form. For the parameters that do not have a closed form solution for the update, we just present the part of the objective function that depends on the concerned parameter and some numeric optimization method has to be used for optimizing the lower bound. Since $\boldsymbol{\phi}_{nm}$ is a multinomial distribution, the updated values of the k components should be normalized to unity. Note that the optimal value of one of the variational parameters depends on the others and, therefore, an iterative optimization is adopted to minimize the lower bound till convergence is achieved.

.4

Equations (6) and (8) present updates for two new parameters. These parameters come from $\mathbb{E}_q(\log\, p(\boldsymbol{w}_{1nl}|f(\boldsymbol{y}_n)))$ and $\mathbb{E}_q(\log\, p(\boldsymbol{z}_{nm}|f(\boldsymbol{\theta}_n)))$ respectively. Both of these integrations do not have analytic solution and hence a first order Taylor approximation is utilized as also done in [5]. A closer inspection of (12) reveals that δ^2 appears in the denominator of the term $\sum_{i=1}^{k}(\mu_{ni} - \epsilon_{ni})^2/\delta^2$ in the objective. Hence, larger values of δ^2 will nullify any effect from $\boldsymbol{\epsilon}_n$ which, in turn, is affected by the observations $\{\boldsymbol{w}_{2nm}\}$ (as is obvious from (14)). On the other hand, if δ^2 is small enough, $\boldsymbol{\epsilon}_n$ can strongly impact the values of $\boldsymbol{\mu}_n$.

3.1.2 Estimation: For estimation, we maximize the optimized lower bound obtained from the variational inference w.r.t the free model parameters $\boldsymbol{\zeta}_0$ (by keeping the variational parameters fixed). The optimal values of the model parameters are presented in equations (5), (7) and (9). Since $\boldsymbol{\beta}_{mi}$ is a multinomial distribution, the updated values of $k^{(m)}$ components should be normalized to unity. However, no closed form of update exists for $\boldsymbol{\sigma}^2$, and a numeric optimization method has to be resorted to. The part of the objective function that depends on $\boldsymbol{\sigma}^2$ is provided in Eq. (11). Once the optimization in M-step is done, E-step starts and the iterative update is continued till convergence. The variational parameters $\{\boldsymbol{\mu}_n\}_{n=1}^N$ are then investigated which serve as proxy for the refined posterior estimates of $\{\boldsymbol{y}_n\}_{n=1}^N$. The main steps of inference and estimation are concisely presented in Algorithm 1.

Algorithm 1 Learning **BC^{3}E**

Input: $\boldsymbol{W}$.
Output: $\boldsymbol{\theta}^m, \{\boldsymbol{\mu}_n\}_{n=1}^N$.

Initialize $\boldsymbol{\theta}^m$, $\{\boldsymbol{\zeta}_n\}_{n=1}^N$.
Until Convergence
E-Step
Until Convergence
1. Update κ_n using Eq. (6) $\forall n \in \{1, 2, \cdots, N\}$.
2. Update ξ_n using Eq. (8) $\forall n \in \{1, 2, \cdots, N\}$.
3. Update ϕ_{nmi} using Eq. (4) $\forall n, m, i$. Normalize $\boldsymbol{\phi}_{nm}$.
4. Maximize (12) w.r.t. $\boldsymbol{\mu}_n\ \forall n$.
5. Maximize (13) w.r.t. $\boldsymbol{\sigma}_n^2\ \forall n$ s.t. $\boldsymbol{\sigma}_n^2 \geq \mathbf{0}$.
6. Maximize (14) w.r.t. $\boldsymbol{\epsilon}_n\ \forall n$.
7. Maximize (10) w.r.t. $\boldsymbol{\delta}_n^2\ \forall n$ s.t. $\boldsymbol{\delta}_n^2 \geq 0$.
M-Step
8. Update $\boldsymbol{\mu}$ using Eq. (5).
9. Update δ^2 using Eq. (9).
10. Update β_{mij} using Eq. (7) $\forall m, i, j$. Normalize $\boldsymbol{\theta}_{mi}$.
11. Maximize (11) w.r.t. $\boldsymbol{\sigma}^2$ s.t. $\boldsymbol{\sigma}^2 \geq \mathbf{0}$.

4 Privacy Preserving Learning

Most of the privacy-aware distributed data mining techniques developed so far have focused on classification or on association rules [4, 11]. There has also been some work on distributed clustering for *vertically partitioned data* (different sites contain different attributes/features of a common set of records/objects) [15], and on parallelizing clustering algorithms for *horizontally partitioned data* (i.e. the objects are distributed amongst the sites, which record the same set of features for each object) [9]. These techniques, however, do not

Update Equations	
$\phi^*_{nmi} \propto \exp\left(\epsilon_{ni} + \sum_{j=1}^{k^{(m)}} \beta_{mij} w_{2nmj}\right)\ \forall n, m, i.$ (4)	$\boldsymbol{\mu}^* = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{\mu}_n.$ (5)
$\kappa^*_n = \sum_{i=1}^{k} \exp(\mu_{ni} + \sigma^2_{ni}/2)\ \forall n.$ (6)	$\beta^*_{mij} \propto \sum_{n=1}^{N} \phi_{nmi} w_{2nmj}\ \forall j \in 1, 2, \cdots, k^m.$ (7)
$\xi^*_n = \sum_{i=1}^{k} \exp(\epsilon_{ni} + \delta^2_{ni}/2)\ \forall n.$ (8)	$\delta^2 = \frac{1}{Nk}\sum_{n=1}^{N}\sum_{i=1}^{k}\left[(\epsilon_{ni} - \mu_{ni})^2 + \sigma^2_{ni} + \delta^2_{ni}\right].$ (9)
$\mathcal{L}_{[\boldsymbol{\delta}^2_n]} = -\frac{1}{2}\sum_{i=1}^{k}\frac{\delta^2_{ni}}{\delta^2} - \frac{1}{2}\sum_{i=1}^{k}\log(\delta^2_{ni}) - \frac{r_2}{\xi_n}\sum_{i=1}^{k}\exp(\epsilon_{ni} + \delta^2_{ni}/2).$ (10)	$\mathcal{L}_{[\boldsymbol{\sigma}^2]} = -\frac{N}{2}\sum_{i=1}^{k}\log(\sigma^2_i) - \frac{1}{2}\sum_{n=1}^{N}\sum_{i=1}^{k}[\frac{\sigma^2_{ni} + (\mu_{ni} - \mu_i)^2}{\sigma^2_i}].$ (11)
$\mathcal{L}_{[\boldsymbol{\mu}_n]} = -\frac{1}{2}\sum_{i=1}^{k}\frac{(\mu_{ni} - \mu_i)^2}{\sigma^2_i} - \frac{1}{2\delta^2}\sum_{i=1}^{k}(\mu_{ni} - \epsilon_{ni})^2 + \sum_{l=1}^{r_1}\sum_{i=1}^{k} w_{1nli}\mu_{ni} - \frac{r_1}{\xi_n}\sum_{i=1}^{k}\exp(\mu_{ni} + \sigma^2_{ni}/2).$ (12)	
$\mathcal{L}_{[\boldsymbol{\sigma}^2_n]} = -\frac{1}{2}\sum_{i=1}^{k}\frac{\sigma^2_{ni}}{\sigma^2_i} - \frac{1}{2}\sum_{i=1}^{k}\log(\sigma^2_{ni}) - \frac{1}{2}\sum_{i=1}^{k}\frac{\sigma^2_{ni}}{\delta^2} - \frac{r_1}{\kappa_n}\sum_{i=1}^{k}\exp(\mu_{ni} + \sigma^2_{ni}/2).$ (13)	
$\mathcal{L}_{[\boldsymbol{\epsilon}_n]} = \sum_{m=1}^{r_2}\sum_{i=1}^{k}\phi_{nmi}\epsilon_{ni} - \frac{1}{\xi_n}\sum_{i=1}^{k}\exp(\epsilon_{ni} + \delta^2_{ni}/2) - \frac{1}{2}\sum_{i=1}^{k}\frac{(\epsilon_{ni} - \mu_{ni})^2}{\delta^2}.$ (14)	

Table 1: Equations for update of variational and model parameters in **BC^{3}E**

specifically address privacy issues, other than through encryption [20].

This is also true of earlier, data-parallel methods [9] that are susceptible to privacy breaches, and also need a central planner that dictates what algorithm runs on each site. Finally, recent works on distributed differential privacy focus on query processing rather than data mining [7].

In the sequel, we show that the inference and estimation in **BC^{3}E** using **VEM** allows solving the cluster ensemble problem in a way that preserves privacy. Depending on how the objects with their cluster/class labels are distributed in different "data sites", we can have three scenarios – i) Row Distributed Ensemble, ii) Column Distributed Ensemble, and iii) Arbitrarily Distributed Ensemble.

4.1 Row Distributed Ensemble: In the row distributed ensemble learning framework, the test set $\mathcal{X}$ is partitioned into D parts and different parts are assumed to be at different locations. The objects from partition d are denoted by $\mathcal{X}_d$ so that $\mathcal{X} = \cup_{d=1}^{D}\mathcal{X}_d$. Now, a careful look at the E-step equations reveal that the update of variational parameters corresponding to each object in a given iteration is independent of those of other objects. Therefore, we can maintain a client-server based framework where the server only updates the model parameters (in the M-step) and the clients (there should be as many number of clients as there are distributed data sites) update the variational parameters.

For instance, consider a situation where a dataset is partitioned into two subsets $\mathcal{X}_1$ and $\mathcal{X}_2$ and these two subsets are located in two different data sites. Data site 1 has access to $\mathcal{X}_1$ and a set of clustering and classification results pertaining to objects belonging to $\mathcal{X}_1$. Similarly, data site 2 has access to $\mathcal{X}_2$ and a set of clustering and classification results corresponding to $\mathcal{X}_2$. Further assume that a set of distributed classification (clustering) algorithms were used to generate the class (cluster) labels of the objects belonging to each set. Now, data site 1 can update the variational parameters $\boldsymbol{\zeta}_n$, $\forall \mathbf{x}_n \in \mathcal{X}_1$. Similarly, data site 2 can update the variational parameters for all objects $\mathbf{x}_n \in \mathcal{X}_2$. Once the variational parameters are updated in the E-step, the server gathers information from two sites and updates the model parameters. Now, a closer inspection of the M-step update equations reveals that each of them contains a summation over the objects. Therefore, individual data sites can send only some collective information to the server without transgressing privacy. For example, consider the update equation for β_{mij}. Eq. (7) can be broken as follows:

$$\beta_{mij}{}^* \propto \sum_{x_n \in \mathcal{X}_1} \phi_{nli} w_{2nli} + \sum_{x_n \in \mathcal{X}_2} \phi_{nli} w_{2nli} \tag{4.15}$$

The first and second terms can be calculated in data sites 1 and 2 separately and sent to the server where the two terms can be added and β_{mij} can get updated $\forall m, i, j$. Similarly, the other M-step update equations (performed by the server in an analogous way) also do

not reveal any information about class or cluster labels of objects belonging to different data sites.

4.2 Column Distributed Ensemble: In the column distributed framework, different data sites share the same set of objects but only a subset of base clusterings or classification results are available to each data site. For example, consider that we have two data sites and four sets of class and cluster labels and each data site has access to only two sets of classification or clustering results. Assume that data site 1 has access to the 1st and 2nd classification and clustering results and data site 2 has access to the rest of the results. As in the earlier case, a single server and two clients (corresponding to two different data sites) are maintained. Since each data site has access to all the objects, it is necessary to share the variational parameters corresponding to these objects. Therefore, $\{\kappa_n, \xi_n, \boldsymbol{\mu}_n, \boldsymbol{\sigma}_n, \boldsymbol{\epsilon}_n, \boldsymbol{\delta}_n\}_{n=1}^N$ are all updated in the server (which is accessible from each client).

The site (and object) specific variational parameters $\{\phi_{nmi}\}$, however, cannot be shared and should be updated in individual sites. This means that the updates (6), (8), (12), (14), (10) and (13) should be performed in the server. On the other hand, the update for $\{\phi_{nmi}\}\forall n, i$ and $m \in \{1, 2\}$ (corresponding to the 1nd and 2nd clustering or classification results) should be performed in data site 1. Similarly, the update for $\{\phi_{nmi}\}\ \forall n, i$ and $m \in \{3, 4\}$ has to be performed in data site 2. However, while updating $\{\boldsymbol{\mu}_n\}$, the calculation of the term $\sum_{l=1}^{r_1}\sum_{i=1}^{k} w_{1nli}\mu_{ni}$ has to be performed without revealing the class labels $\{\boldsymbol{w}_{1nl}\}$ to the server. To that end, it can be rewritten as:

(4.16)
$$\sum_{l=1}^{r_1}\sum_{i=1}^{k} w_{1nli}\mu_{ni} = \sum_{l=1}^{2}\sum_{i=1}^{k} w_{1nli}\mu_{ni} + \sum_{l=3}^{4}\sum_{i=1}^{k} w_{1nli}\mu_{ni},$$

where the first term can be computed in data site 1 and the second term can be computed by data site 2 and then can be added in the server. It can be seen that $\{\boldsymbol{w}_{1nl}\}$ can never be recovered by the server and hence privacy is ensured in the updates of the E-step. Except for $\{\beta_{mij}\}$, all other model parameters can be updated in the server in the M-step. However, the parameters $\{\beta_{mij}\}$ have to be updated separately inside the clients. Since $\{\beta_{mij}\}$ do not appear in any update equation performed in the server, there is no need to send these parameters to the server either. Therefore, in essence, the clients update the parameters $\{\phi_{nmi}\}$ and $\{\beta_{mij}\}$ in E-step and M-step respectively, and the server updates the remaining parameters.

4.3 Arbitrarily Distributed Ensemble: In an arbitrarily distributed ensemble, each data site has access to only a subset of the data points or a subset of the classification and clustering results. Fig. 3 shows a situation with arbitrarily distributed ensemble with six data sites.

We now refer to Fig. 4 and explain the privacy preserved EM update for this setting. As before, corresponding to each different data site, a client node is created. Clients that share a subset of the objects should have access to the variational parameters corresponding to common objects. To highlight the sharing of objects by clients, the test set $\mathcal{X}$ is partitioned into four subsets — $\mathcal{X}_1, \mathcal{X}_2, \mathcal{X}_3$ and$\mathcal{X}_4$ as shown in Fig. 3. Similarly, the columns are also partitioned into three subsets: G_1, G_2, and G_3.

Now, corresponding to each row partition, an "Auxiliary Server"(AS) node is created. Each AS updates the variational parameters corresponding to a set of shared objects. For example, in Fig. 4.3, AS_1 updates the variational parameters corresponding to $\mathcal{X}_1$ (using equations (8), (6), (12), (13), (14), and (10)). However, any variational parameter that is specific to both an object and a column is updated separately inside the corresponding client (and hence it is connected with C_1 and C_2). Therefore, $\{\phi_{nmi} : n \in \mathcal{X}_1, m \in G_1\}$ are updated inside client 1 and $\{\phi_{nmi} : n \in \mathcal{X}_1, m \in G_2 \cup G_3\}$ are updated inside client 2 (using Eq. (4)). Once all variational parameters are updated in the E-step, M-step starts. Corresponding to each column partition, an "Auxiliary Client" (AC) node is created. This node updates the model parameters β_{mij} (using Eq. (7)) which are specific to columns belonging to G_1. Since C_1, C_3, and C_5 share the columns from the subset G_1, AC_1 is connected with these three nodes in Fig. 4.3. The remaining model parameters are, however, updated in a "Server" (using equations (5), (9), (11)).

In Fig. 4.3, the bidirectional edges indicate that messages are sent to and from the connecting nodes. We have avoided separate arrows for each direction only to keep the figure uncluttered. The edges are also numbered near to their origin. For a comprehensive understanding of the privacy preservation, the messages transfered through each edge have also been enlisted in the supplementary material. The messages sent from the auxiliary servers to the main server are of the form given in Eq. (4.15) and are denoted as "partial sums". Expectedly, messages sent out from a client node are "masked" in such a way that no other node can decode the cluster labels or class labels of points belonging to that client. This approach is completely general and will work for any arbitrarily partitioned ensemble given that each partition contains at least two sets of classification

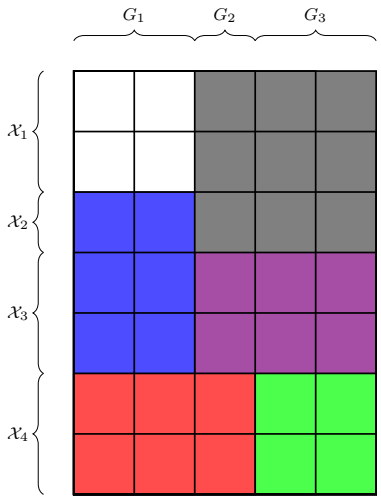

Figure 3: Arbitrarily Distributed Ensemble

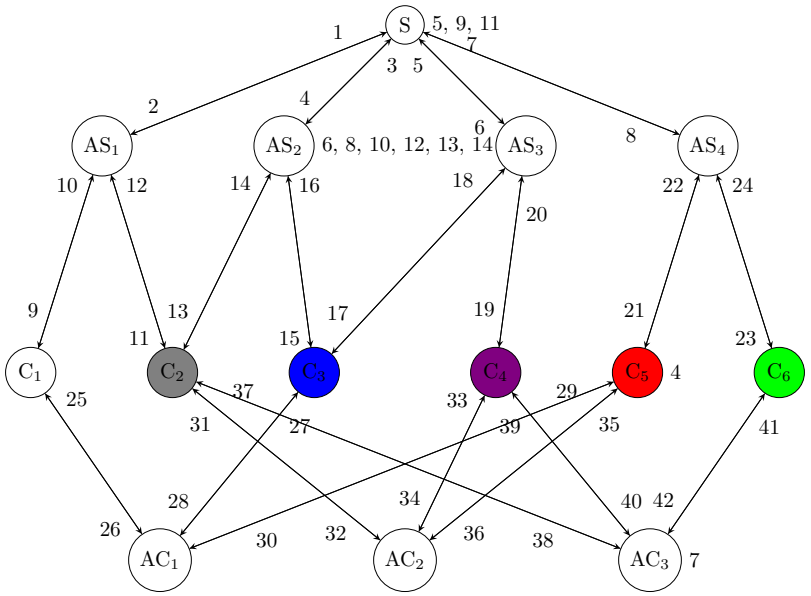

Figure 4: Parameter Update for Arbitrarily Distributed Ensemble

results. Note that the ACs and ASs are only helpful in conceptual understanding of the parameter update and sharing. In practice, there is no real need for these extra storage devices/locations. Client nodes can themselves take the place of ASs and ACs and even the main server as long as the updates are performed in proper sequence[1].

5 Experiments

In this section, two different sets of experiments are reported. The first set is for transfer learning with a text classification data from eBay Inc. The other set is for non-transductive semisupervised learning where some publicly available datasets are used to simulate the working environment of **BC^{3}E**.

5.1 Transfer Learning: To show the capability of **BC^{3}E** in solving transfer learning problems, we use a large scale text classification dataset from eBay Inc. The training data consists of 83 million items sold over a three month period of time and the test set contains several millions of items sold a few days after the training period. More details about the dataset can be found in [18]. eBay organizes items into a six-level category structure where there are 39 top level nodes called *meta categories* and 20K+ bottom level nodes called *leaf categories*. The dataset is generated when users provide the titles of items they intend to sell on eBay. Each title is limited to 50 characters, based on which the user gets recommendation of some *leaf categories* the item should belong to. Such categorization of the item helps a seller list an item in the correct branch of the product list, thereby allowing a buyer more easily search through a list of few million items sold via eBay every single day. A carefully designed k-Nearest Neighbor (k-NN) classifier (with the help of improved search engine algorithms) categorizes each of the items in less than 100 ms [18]. However, due to the large number of categories (20K), items belonging to similar types of categories often get misclassified.

To avoid such confusion, larger categories are formed by aggregating examples from categories which are relatively difficult to separate. Such aggregation is easy once the confusion matrix of the classification, obtained from a development dataset, is partitioned and strongly connected vertices (each vertex representing one of 20K *leaf categories*) are identified from the confusion graph, thereby forming a set of cliques which represent the large categories. Note that the large categories so discovered might not at all follow the internal hierarchy that is maintained. Next, clustering is performed with examples belonging to each of the large categories and the clustering results, along with the predictions from k-NN classification, are fed to **BC^{3}E** (and also to its competitors *i.e.* **C^{3}E**, **BGCM**, and **LWE**). The idea here is to first reduce the classification space and then use unsupervised information to refine the predictions from k-NN on a smaller number of categories. The

[1]Note that such framework allows running the updates of the same stage in parallel in different sites, thereby saving the computation time in large scale implementations.

number of *leaf categories* belonging to such large categories usually varies between 4-10.

However, the dataset is very dynamic and, typically over a span of three months, 20% of new words are added to the existing vocabulary. One can retrain the existing k-NN classifier every three months, but the training process requires collecting new labeled data which is time consuming and expensive. One can additionally design classifiers to segregate examples belonging to each of the large categories. However, such approach might not improve much upon the performance of the initial k-NN classifier if the data changes so frequently. Therefore, we require a system that can adaptively predict newer examples without retraining the existing classifier or employing another set of classification algorithms. **BC^{3}E** is useful in such settings. The parameter δ can adjust the weights of prediction from classifiers and unsupervised information. As the results reported in Table 5.1 reveal, as long as the classification performance is not that poor, **BC^{3}E** can improve on the performance of k-NN using the clustering ensemble.

The column "Group ID" denotes anonymized groups representing different large categories. $|\mathcal{X}|$ shows the number of examples in the test data. The column "C^{3}E-Ideal" shows the performance of **C^{3}E** if the correct tuning parameter for **C^{3}E** were known. For a transfer learning problem, estimating such tuning parameter requires some labeled data from the target set which is not available in our setting. If the tuning parameter is chosen from cross-validation on the training data, the final prediction on target set can get affected adversely if the underlying distribution changes (and in fact it does in our experiments). Therefore, we need to adopt a fail-safe approach where we can do at least as good as the k-NN prediction. The results reveal that **BC^{3}E** significantly outperforms **BGCM** and **LWE**, and sometimes achieves as good a performance as **C^{3}E**-Ideal (*i.e.* when correct tuning parameter of **C^{3}E** is known). The performance of **C^{3}E**-Ideal can essentially be considered as the best accuracy one could achieve from the given inputs (*i.e.* class and cluster labels) using other existing algorithms — **BGCM**, **LWE**, **C^{3}E** — that work on the same design space. Though **BGCM** has a tuning parameter, its variation did not affect performance much and we just report results corresponding to unity value of this parameter.

5.2 Semi-supervised Learning: Six datasets are used in our experiments for semi-supervised learning: *Half-Moon* (a synthetic dataset with two half circles representing two classes), *Circles* (another synthetic dataset that has two-dimensional instances that form two concentric circles — one for each class), and four datasets from the *Library for Support Vector Machines* — *Pima Indians Diabetes*, *Heart*, *German Numer*, and *Wine*. In order to simulate semi-supervised settings where there is a very limited amount of labeled instances, small percentages (see the values reported in Table 5.2) of the instances are randomly selected for training, whereas the remaining instances are used for testing (target set). We perform 20 trials for every dataset. For running experiments with **BGCM**, and **C^{3}E**, the parameters reported in [13] and [2] are used respectively. The parameters of **BC^{3}E** are initialized randomly and approximately 10 EM iterations are enough to get the results reported in Table 5.2. The classifier ensemble consists of decision tree (C4.5), linear discriminant, and generalized logistic regression. Cluster ensembles are generated by means of multiple runs of k-means [2]. **LWE** [12] is better suited for transfer learning applications and hence has been left out from comparison. The column "Best" in Table 5.2 refers to the performance of the best classifier in the ensemble. Note that **BC^{3}E** has superior performance for the most difficult problems, where one has an incentive to use a more complex mechanism. Most importantly, **BC^{3}E** has the privacy preserving property not present in any of its counterparts.

6 Conclusion and Future Work

The **BC^{3}E** model proposed in this paper has been shown to be useful for difficult non-transductive semisupervised and transfer learning problems. A good tradeoff between accuracy and privacy has also been established empirically – a property absent in any of **BC^{3}E**'s competitors. With minor modification, **BC^{3}E** can also handle soft outputs from classification and clustering ensembles which can further improve the results.

References

[1] A. Acharya, E.R. Hruschka, and J. Ghosh, *A privacy-aware bayesian approach for combining classifier and cluster ensembles*, in SocialCom/PASSAT, 2011, pp. 1169–1172.

[2] A. Acharya, E.R. Hruschka, J. Ghosh, and S. Acharyya, *An optimization framework for semi-supervised and transfer learning using multiple classifiers and clusterers*, CoRR, abs/1206.0994 (2012).

[3] A. Acharya, E. R. Hruschka, J. Ghosh, and S. Acharyya, *C^{3}E: A Framework for Combining Ensembles of Classifiers and Clusterers*, in 10th Int. Workshop on MCS, 2011.

[4] D. Agrawal and C. C. Aggarwal, *On the design and quantification of privacy preserving data mining algorithms*, in Symposium on Principles of Database Systems, 2001.

Group ID	$\lvert\mathcal{X}\rvert$	k-NN	BGCM	LWE	C^{3}E-Ideal	BC^{3}E
42	1299	64.90	73.78 (± 0.94)	76.86 (± 1.01)	83.99 (± 0.41)	83.68 (± 1.09)
84	611	63.67	69.23 (± 0.17)	75.24 (± 0.26)	81.18 (± 0.16)	76.27 (± 1.31)
86	2381	77.66	84.33 (± 2.74)	83.29 (± 1.02)	92.78 (± 0.35)	87.20 (± 0.91)
67	789	72.75	72.75 (± 0.07)	78.03 (± 0.72)	82.64 (± 0.82)	81.75 (± 1.37)
52	1076	76.95	77.01 (± 1.18)	77.49 (± 1.41)	88.38 (± 0.22)	85.04 (± 2.14)
99	827	84.04	85.12 (± 0.52)	86.90 (± 0.92)	91.54 (± 0.27)	91.17 (± 0.82)
48	3445	86.33	86.19 (± 0.25)	90.38 (± 1.03)	92.71 (± 0.31)	92.71 (± 1.16)
94	440	79.32	81.08 (± 0.73)	82.52 (± 0.83)	85.45 (± 0.09)	85.45 (± 0.79)
35	4907	82.41	82.10 (± 0.37)	85.08 (± 1.39)	88.16 (± 0.17)	88.22 (± 1.21)
45	1952	74.80	73.12 (± 0.81)	73.64 (± 1.68)	84.32 (± 0.23)	77.97 (± 0.47)

Table 2: Performance of **BC^{3}E** on text classification data — Avg. Accuracies ±(Standard Deviations).

Dataset (% of tr. data)	$\lvert\mathcal{X}\rvert$	Ensemble	Best	BGCM	C^{3}E	BC^{3}E
Half-moon(2%)	784	92.53(±1.83)	93.02(±0.82)	92.16(±1.47)	**99.64**(±0.08)	98.23(±2.03)
Circles(2%)	1568	60.03(±8.44)	95.74(±5.15)	78.67(±0.54)	**99.61**(±0.83)	97.91(±0.74)
Pima(2%)	745	68.16(±5.05)	69.93(±3.68)	69.21(±4.83)	70.31(±4.44)	**72.83**(±0.49)
Heart(7%)	251	77.77(±2.55)	79.22(±2.20)	82.78(±4.82)	**82.85**(±5.25)	82.53(±1.14)
G. Numer(10%)	900	70.96(±1.00)	70.19(±1.52)	73.70(±1.06)	74.44(±3.44)	**74.61**(±1.62)
Wine(10%)	900	79.87(±5.68)	80.37(±5.47)	75.37(±13.66)	**83.62**(±6.27)	82.20(±1.07)

Table 3: Comparison of **BC^{3}E** with **C^{3}E** and **BGCM** — Avg. Accuracies ±(Standard Deviations).

[5] D.M. Blei and J.D. Lafferty, *A correlated topic model of science*, Annals of Applied Statistics, 1 (2007), pp. 17–35.

[6] A. Blum, *On-line algorithms in machine learning*, in Online Algorithms: The State of the Art, Fiat and Woeginger, eds., LNCS Vol.1442, Springer, 1998.

[7] R. Chen, A. Reznichenko, P. Francis, and J. Gehrke, *Towards statistical queries over distributed private user data*, in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, NSDI'12, 2012.

[8] H. A. Chipman, E. I. George, and R. E. McCulloch, *Bayesian ensemble learning*, in Proc. of Neural Information Processing Systems, 2006, pp. 265–272.

[9] I. S. Dhillon and D. S. Modha, *A data-clustering algorithm on distributed memory multiprocessors*, in Proc. Large-scale Parallel Knowledge Discovery and Data Mining Systems Workshop, ACM SIGKnowledge Discovery and Data Mining, August 1999.

[10] N. U. Edakunni and S. Vijayakumar, *Efficient online classification using an ensemble of bayesian linear logistic regressors*, in 8th Int. Workshop on MCS, 2009, pp. 102–111.

[11] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke, *Privacy preserving mining of association rules*, in Knowledge Discovery and Data Mining, 2002.

[12] J. Gao, W. Fan, J. Jiang, and J. Han, *Knowledge transfer via multiple model local structure mapping*, in Proc. of KDD, 2008, pp. 283–291.

[13] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, *Graph-based consensus maximization among multiple supervised and unsupervised models*, in Proc. of NIPS, 2009, pp. 1–9.

[14] Z. Ghahramani and H. Kim, *Bayesian classifier combination*, tech. report, 2003.

[15] E. Johnson and H. Kargupta, *Collective, hierarchical clustering from distributed, heterogeneous data*, in Large-Scale Parallel Knowledge Discovery and Data Mining Systems, vol. 1759 of LNCScience, 1999, pp. 221–244.

[16] S. Merugu and J. Ghosh, *Privacy perserving distributed clustering using generative models*, in Proc. of ICDM, Nov, 2003, pp. 211–218.

[17] N. C. Oza and K. Tumer, *Classifier ensembles: Select real-world applications*, Inf. Fusion, 9 (2008), pp. 4–20.

[18] D. Shen, J. Ruvini, and B. Sarwar, *Large-scale item categorization for e-commerce*, in CIKM, Oct. 2012.

[19] D. L. Silver and K. P. Bennett, *Guest editor's introduction: special issue on inductive transfer learning*, Machine Learning, 73 (2008), pp. 215–220.

[20] J. Vaidya and C. Clifton, *Privacy-preserving k-means clustering over vertically patitioned data*, in KDD, 2003, pp. 206–215.

[21] H. Wang, H. Shan, and A. Banerjee, *Bayesian cluster ensembles*, Statistical Analysis and Data Mining, 1 (2011), pp. 1–17.